QÜESTIIÓ, vol. 26, 1-2, p. 141-171, 2002

ON SUPERLINEAR MULTIPLIER UPDATE METHODS FOR PARTIAL AUGMENTED LAGRANGIAN TECHNIQUES[*]

E. MIJANGOS[*]

The minimization of a nonlinear function with linear and nonlinear constraints and simple bounds can be performed by minimizing an augmented Lagrangian function, including only the nonlinear constraints. This procedure is particularly interesting in case that the linear constraints are flow conservation equations, as there exist efficient techniques to solve nonlinear network problems. It is then necessary to estimate their multipliers, and variable reduction techniques can be used to carry out the successive minimizations. This work analyzes the possibility of estimating these multipliers using Newton-like methods. Several procedures are put forward which combine first and second-order estimation, and are compared with each other and with the Hestenes-Powell multiplier estimation by means of computational tests.

Keywords: Partial Augmented Lagrangian, Variable Reduction, Lagrange Multipliers, Superlinear-order Estimation, Nonlinear Network

AMS Classification (MSC 2000): 90C30, 90C35

[*] Department of Applied Mathematics and Statistics and O.R., Zientzi Fakultatea (UPV/EHU), P.O. Box 644, 48080-Bilbao, Spain. Fax 34 944648500, e-mail: mepmifee@lg.ehu.es.

[*] Partially supported by CICYT grant TAP1999-1075-C0201.

– Received Setember 2000.
– Accepted November 2001.

1. INTRODUCTION AND MOTIVATION

Consider the linearly and nonlinearly constrained problem **(EGP)**:

$$\begin{aligned} &\text{minimize} && f(x) && (1)\\ &\text{subject to:} && Ax = b && (2)\\ & && c(x) = 0 && (3)\\ & && l \le x \le u, && (4) \end{aligned}$$

where

1. $f : \mathbb{R}^n \longrightarrow \mathbb{R}$. $f(x)$ is nonlinear and twice continuously differentiable on the feasible set defined by constraints (2) and (4).
2. A is an $m \times n$ matrix and b an m-vector.
3. $c : \mathbb{R}^n \longrightarrow \mathbb{R}^r$, is such that $c = [c_1, \cdots, c_r]^t$, $c_i(x)$ being linear or nonlinear and twice continuously differentiable on the feasible set defined by constraints (2) and (4) $\forall i = 1, \cdots, r$.

Throughout this work the gradient of f at x is defined as the column vector $\nabla f(x)$, and matrix $\nabla c(x) = [\nabla c_1(x), \cdots, \nabla c_r(x)]$ is the transpose of the *Jacobian* of c at x, though here, for convenience, it will simply be called Jacobian.

The idea is to solve **EGP** by solving a series of approximate problems whose solutions «converge» on the solution of the original problem in a well-defined sense.

In practice, it is only necessary to solve a finite number of approximate problems to arrive at an acceptable approximate solution to the original problem.

Real problems with this same structure exist and have a high dimensionality (e.g., when A is a node-arc incidence matrix). They often need to be solved and it is important to find the procedure that will solve them with the highest efficiency. The short-term hydroelectric power generation optimization problem [5, 32] and the short-term hydrothermal coordination of electric power generation [16] are examples of this type.

One possible way to address problem **EGP** would be to use of Projected Lagrangian methods [12]. This would lead to solution of a series of subproblems of the type:

$$\begin{aligned} &\text{minimize} && \Phi^k(x)\\ &\text{subject to:} && Ax = b\\ & && l \le x \le u\\ & && \nabla c(x^k)^t x = -c(x^k) + \nabla c(x^k)^t x_k \end{aligned}$$

where $\Phi^k(x)$ is either a quadratic approximation to the partial Lagrangian function, or a nonlinear function based on the partial augmented Lagrangian, as the well-known MINOS 5.5 package [27] does. When A is a node-arc incidence matrix this subproblem is a nonlinear network flow problem with linear side constraints, which could be solved by specialized techniques of primal partitioning for network flow problems [19]. This possibility has already been explored [15].

In this work **EGP** is solved using partial augmented Lagrangian techniques [1, 2, 3, 6, 30], as is done in [4, 7, 21, 22, 23, 24], where only the general constraints (3) are included in the Lagrangian.

There is a class of problems for which these techniques have potential interest, in particular those that incorporate *flow conservation equations* as linear constraints (2), as there are efficient techniques for solving nonlinear network problems, see [9, 31]. In [21, 22, 23, 24] the constraints (3) are the *side constraints* of a nonlinear network flow problem. The application of these techniques consists of two fundamental steps. The first is the solution of problem **(EGS)**:

$$\begin{array}{lll} & \underset{x}{\text{minimize}} & L_\rho(x,\mu) \\ (5) & \text{subject to} & Ax = b \\ & & l \le x \le u, \end{array}$$

where $\rho \in \mathbb{R}$, such that $\rho > 0$, and $\mu \in \mathbb{R}^r$ are fixed,

$$L_\rho(x,\mu) = f(x) + \mu^t c(x) + \frac{1}{2}\rho c(x)^t c(x).$$

Should the solution $\tilde{x}$ obtained by solving **EGS** be infeasible with respect to (3), the second step, which is the updating of the estimation μ of the Lagrange multipliers of constraints (3), is carried out —also updating, if necessary, the penalty coefficient ρ— followed by a return to the first step. Should $\tilde{x}$ be feasible (or the violation of constraints (3) sufficiently small) the procedure ends.

It is of paramount importance that the estimation of multipliers μ be as accurate as possible, otherwise the convergence of the algorithm can be severely handicapped, as shown in [2].

In practice there are two first-order procedures to estimate μ. On the one hand the method put forward by Hestenes [18] and Powell [29]

$$\tilde{\mu} = \mu + \rho c(\tilde{x}),$$

and on the other hand μ_L obtained through the classical solution to the system of Kuhn-Tucker necessary conditions

$$(6) \qquad \nabla f(\tilde{x}) + \nabla c(\tilde{x})\mu + A^t\pi + \lambda = 0$$

by least squares, as suggested in [12]. A study of the viability of using these multiplier estimation techniques together with the solution of problem **EGS** (5) by the Murtagh and Saunders's active set technique [26] was carried out in [21], using the method suggested by Hestenes & Powell, and in [24], by solving the Kuhn-Tucker necessary conditions using least squares.

An interesting alternative to these methods is the second-order multiplier estimation for inexact minimization suggested by Tapia in [30], and whose convergence analysis is due to Bertsekas in [2].

In this paper the applicability of the Tapia's second-order multiplier estimate is analyzed when subproblem (5) is solved using variable reduction techniques. As a consequence, under variable reduction techniques an algorithm that is a combination of superlinear and first-order multiplier methods is designed to improve the results obtained using first-order estimates over large-scale nonlinear network problems with side constraints [21, 24]. In the same algorithm, it is shown how to compute the superlinear-order multiplier estimate efficiently by means of suitable matrix computations. Finally, these techniques are extended to problem **(IGP)**:

$$\begin{aligned}
&(7) && \text{minimize} && f(x)\\
&(8) && \text{subject to} && Ax = b\\
&(9) && && \alpha \leq c(x) \leq \beta\\
&(10) && && l \leq x \leq u,
\end{aligned}$$

using the results of the work by Mijangos and Nabona in [21], which is based on the theory developed by Bertsekas (1982) in [2] for extending the multiplier methods obtained for equality constrained problems to one-sided inequality constrained problems, without need of using slack variables. Numerical tests over nonlinear network problems with side constraints are carried out, several algorithmic alternatives being compared with each other and with the Hestenes-Powell multiplier estimation.

The paper is organised as follows. Section 2 analyzes the applicability of these techniques to solve problem **EGP** when subproblem (5) is solved by the Murtagh and Saunders procedure [26]. Also, in Section 2, an algorithm combining superlinear and first-order multiplier estimate techniques is put forward and various practical issues are analyzed. Section 3 extends these techniques to problem **IGP**. Section 4 offers some details about the implementation. Section 5 presents computational tests. Finally, Section 6 offers the conclusions.

2. APPLICABILITY OF THE NEWTON-TYPE METHODS

There follows a new analysis of the applicability of the inexact Newton-like methods in combination with variable reduction techniques.

Let us consider any update formula U of μ in the solution of **EGP** by means of multiplier methods, where the subproblem to solve is (5). The first-order optimality conditions to be fulfilled by the optimizer $(x^*, \mu^*, \pi^*, \lambda^*)$ at the end of this sequential procedure, for all $\rho > 0$ large enough, give rise to the following system

$$\nabla_x L_\rho(x,\mu) + A^t\pi + \lambda = 0 \tag{11}$$
$$c(x) = 0 \tag{12}$$
$$Ax = b, \tag{13}$$

where
$$\nabla_x L_\rho(x,\mu) = \nabla f(x) + \nabla c(x)[\mu + \rho c(x)].$$

Throughout this work we assume that $(x^*, \mu^*, \pi^*, \lambda^*)$ is an optimizer at the end of the sequential procedure satisfying the strict complementarity condition.

Likewise, we assume that given a pair (μ, ρ), ρ is greater than a certain threshold (see Lemma 1.25 in [2]) in order to make sure the full rank of the reduced Hessian of the augmented Lagrangian in the subproblem **EGS**, see (5). Since this work is concerned with the solution of problems where the aforementioned reduced Hessian is dense in general, the highest dimension of this matrix is limited to 500×500.

Let be $\widetilde{x} = x(\mu, \rho)$ an optimizer of this subproblem. It is obvious that the only information we have about the active variables at x^* and the bound where they are fixed is given by the solution of subproblem **EGS**.

Newton's method for solving the system consisting of (11–13) and the equations of the active variables at $\widetilde{x}$ involves solving the linear system

$$\begin{array}{|cc|c|c|c|}
\multicolumn{2}{c}{n} & \multicolumn{1}{c}{r} & \multicolumn{1}{c}{m} & \multicolumn{1}{c}{\widehat{t}} \\
\hline
\multicolumn{2}{|c|}{\nabla^2_{xx} L_\rho(\widetilde{x},\mu)} & \nabla c(\widetilde{x}) & A^t & \begin{array}{c} 0 \\ \hline \mathbb{1} \end{array} \\
\hline
\multicolumn{2}{|c|}{\nabla c(\widetilde{x})^t} & & & \\
\cline{1-2}
\multicolumn{2}{|c|}{A} & \multicolumn{3}{c|}{0} \\
\cline{1-2}
0 & \multicolumn{1}{|c|}{\mathbb{1}} & & & \\
\hline
\end{array}
\begin{array}{|c|}
\hline
\Delta x \\
\hline
\Delta \mu \\
\hline
\Delta \pi \\
\hline
\Delta \lambda \\
\hline
\end{array}
=
\begin{array}{|c|}
\hline
-\nabla_x L_\rho(\widetilde{x},\mu) \\
-A^t\pi - \lambda \\
\hline
-c(\widetilde{x}) \\
\hline
0 \\
\hline
0 \\
\hline
\end{array}, \tag{14}$$

where $\widehat{t}$ stands for the number of variables fixed at their bounds in the solution of subproblem (5).

Let $\nabla^2\mathcal{L}$ denote the matrix of coefficients of (14), which should be nonsingular. Let us assume that matrix $\nabla^2_{xx} L_\rho(\widetilde{x},\mu)$ is nonsingular and A has full row rank; then, the necessary and sufficient condition for matrix $\nabla^2\mathcal{L}$ to be nonsingular is that

$$\widehat{A}(\widetilde{x}) = \begin{array}{|c|c|c|l}\hline \nabla c_B(\widetilde{x})^t & \nabla c_S(\widetilde{x})^t & \nabla c_N(\widetilde{x})^t & \}r \\ \hline B_A & S_A & N_A & \}m \\ \hline \multicolumn{2}{|c|}{0} & \mathbb{1} & \}\widehat{t} \\ \hline \end{array} \tag{15}$$

has full row rank, where the second row section contains the matrix A partitioned into three submatrices, B_A and S_A corresponding to the last basic and superbasic submatrices after using the Murtagh and Saunders method to solve subproblem **EGS** (5), N_A being the columns of A corresponding to the $\widehat{t}$ active variables in $\widetilde{x}$. The partition of $\nabla c(\widetilde{x})^t$ is the same as that of A. The variables associated with columns N_A are precisely those variables whose bound is predicted to be active at the optimizer. However, it could happen that the submatrix

$$BS = \begin{array}{|c|c|}\hline \nabla c_B(\widetilde{x})^t & \nabla c_S(\widetilde{x})^t \\ \hline B_A & S_A \\ \hline \end{array}$$

does not have full row rank, which could be due to one of two reasons:

- either $\nabla c(\widetilde{x})^t$ does not have full row rank in spite of $\nabla c(x^*)^t$ having it;
- or $\nabla c(\widetilde{x})^t$ has full row rank, but $[\nabla c_B(\widetilde{x})^t \quad \nabla c_S(\widetilde{x})^t]$ does not have it, because the prediction about the variables whose bound will be active at the optimizer is not totally correct.

Both cases can occur if $\widetilde{x}$ is not sufficiently close to x^*.

In the second case we can know (or hope) that $\nabla c(\widetilde{x})^t$ has full row rank either because $\widetilde{x}$ is already close enough to the optimizer, or due to the structure of the problem, or because the rank is checked explicitly. Then, it may be interesting to enlarge BS by adding columns of the submatrix

$$N = \begin{array}{|c|}\hline \nabla c_N(\widetilde{x})^t \\ \hline N_A \\ \hline \end{array} \tag{16}$$

until a matrix is obtained with full row rank. However, this would mean excluding these variables from being bound active variables and forcing their λ^* multiplier to be zero, against the prediction in $\widetilde{x}$. If these variables were not suitably chosen (assuming that «suitable» ones exist), it could happen that the Δx displacement associated with them would not be feasible, and thus all the computational effort expended in solving system (14) would have been lost. Therefore, when $\widetilde{x}$ is sufficiently close to x^*, and in order to complete the row rank of BS (if necessary), we suggest enlarging BS by adding the columns of N whose associated active variables have a multiplier estimate zero or almost zero, see Appendix A in [24].

2.1. Multiplier update in problem EGP

In practice, one does not need to solve system (14). Let $\overline{Z}$ be the following matrix:

$$\overline{Z} = \begin{bmatrix} Z_A^t & 0 \\ 0 & \mathbb{1} \end{bmatrix}, \tag{17}$$

where the unit matrix has dimension $r+m+\widehat{t}$ and Z_A is the variable reduction matrix:

$$Z_A = \begin{bmatrix} -B_A^{-1}S_A \\ \mathbb{1} \\ 0 \end{bmatrix}, \tag{18}$$

such that S_A is enlarged with columns of N_A if it is necessary for BS to have full row rank.

Premultiplying both sides of (14) by $\overline{Z}$ and, since there is a vector Δx_S and only one such that

$$\Delta x = Z_A \Delta x_S, \tag{19}$$

using this last expression we are led from system (14) to the *reduced Newton system*:

$$\begin{array}{cc} s & r \\ \left[\begin{array}{c|c} Z_A^t \nabla_{xx}^2 L_\rho(\widetilde{x},\mu) Z_A & Z_A^t \nabla c(\widetilde{x}) \\ \hline \nabla c(\widetilde{x})^t Z_A & 0 \end{array}\right] \end{array} \left[\begin{array}{c} \Delta x_S \\ \hline \Delta\mu \end{array}\right] = \left[\begin{array}{c} -Z_A^t \nabla_x L_\rho(\widetilde{x},\mu) \\ \hline -c(\widetilde{x}) \end{array}\right]. \tag{20}$$

Since $\nabla^2 \mathcal{L}$ is nonsingular and Z_A has full column rank, the coefficient matrix of system (20) is also nonsingular, hence this system has a single solution $(\Delta x_S, \Delta\mu)$.

By first calculating $\Delta\mu$ we find that for problem **EGP** the U_{INW} update is given by

$$\overline{\mu} = \mu + [J^t H^{-1} J]^{-1} [c(\widetilde{x}) - J^t H^{-1} (Z_A^t \nabla_x L_\rho(\widetilde{x},\mu))] \tag{21}$$

where $J = Z_A^t \nabla c(\widetilde{x})$ is the *reduced Jacobian* in $\widetilde{x}$ and H is either the exact reduced Hessian of the augmented Lagrangian —i.e., $Z_A^t \nabla_{xx}^2 L_\rho(\widetilde{x}, \mu) Z_A$— or an approximation of it (e.g., quasi-Newton or discrete Newton).

We should not forget that J must have full column rank so that the system matrix of (20) will be nonsingular and, in consequence, will have an inverse.

2.2. Solution of the reduced Newton system

The procedure that we are going to describe makes it possible to *switch* from the first-order update to a superlinear-order update «when the circumstances allow and suggest it». The final aim of this combination is to increase its convergence rate as far as possible, at the same time keeping the global convergence of the multiplier method, see [2].

The rules considered in the previous section are complemented with others added in order to check the superlinear convergence at the end of the algorithm. Moreover, three possibilities are considered in relation to the Hessian H (here reduced Hessian) of the augmented Lagrangian, corresponding to the following cases:

- it is the exact Hessian at x^k,
- it is its approximation by finite differences, or
- it is a quasi-Newton approximation (e.g., BFGS).

Let $x^k = x(\mu^k, \rho^k)$ be the solution of the subproblem **EGS** for pair (μ^k, ρ^k), see (5).

As a consequence of §2.1, if for $x = x^k$ the submatrices of the coefficient matrix and the right-hand side of system (20) are denoted as follows:

$$\begin{aligned} H_k &= Z_A^t \nabla_{xx}^2 L_{\rho^k}(x^k, \mu^k) Z_A \\ J_k &= Z_A^t \nabla c(x^k) \\ h_k &= Z_A^t \nabla_x L_{\rho^k}(x^k, \mu^k), \end{aligned}$$

the solution of (20) at x^k is given by the initial calculation of

$$\Delta\mu = (J_k^t H_k^{-1} J_k)^{-1} (c(x^k) - J_k^t H_k^{-1} h_k),$$

μ^{k+1} being calculated by means of

$$\mu^{k+1} = \mu^k + \Delta\mu,$$

then $\overline{h}_k = Z_A^t \nabla_x L_{\rho^k}(x^k, \mu^{k+1})$ is obtained, and finally

$$\Delta x_S = -H_k^{-1}\overline{h}_k, \tag{22}$$

from which the remaining components of Δx are obtained by using expression (19). This Δx may be used in the first iteration of the solution of the new **EGS** subproblem (see (5)) as search direction d, provided that it is feasible with regard to the simple bounds of subproblem **EGS**. However, the reason for this section is to compute the superlinear-order update of μ, when this is possible, and using it if it is reliable enough. Therefore, assuming that H_k is either the exact projected Hessian, or a good enough approximation of it, and that we have a Cholesky factorization $H_k = R_k^t R_k$, the following algorithm is designed. Let $D_k = J_k^t H_k^{-1} J_k$ and $U_O(x, \mu, \rho) = \mu + \rho c(x)$.

Algorithm 1

Step 1. Compute the first-order multiplier estimation U_O and obtain a projected Jacobian J_k having full column rank. Set

$$\widetilde{\mu} = U_O(x^k, \mu^k, \rho^k).$$

If T_{nw0} (*activation rule*) (see §2.3) a projected Jacobian J_k having full column rank is obtained, if it exists, by means of the method suggested in Appendix A of [24]. If either T_{nw0} is not true, or J_k does not exist in spite of this rule being true, the algorithm finishes with the update U_O by default.

Step 2. Compute Cholesky factorization $D_k = \overline{R}_k^t \overline{R}_k$.

(i) The matrix M_k is computed by calculating its columns $M_k^{(i)}$, for $i = 1, \ldots, r$, which is done by successively solving system

$$R_k^t M_k^{(i)} = J_k^{(i)},$$

where $J_k^{(i)}$ denotes the i-th column of J_k.

(ii) The QR factorization of M_k is carried out by obtaining an upper triangular matrix $\overline{R}_k$ such that

$$\begin{bmatrix} \overline{R}_k \\ 0 \end{bmatrix} = Q_k M_k,$$

Q_k being a suitable orthogonal matrix.

Step 3. Compute $\overline{c}_k = c(x^k) - J_k^t H_k^{-1} h_k$.

(i) Solve $R_k^t R_k p = h_k$.

(ii) Compute $\overline{c}_k = c(x^k) - J_k^t p$.

Step 4. Solve $D_k \Delta\mu = \overline{c}_k$. By solving $\overline{R}_k^t \overline{R}_k \Delta\mu = \overline{c}_k$.

Step 5. Compute the superlinear-order update of μ. Set

$$\mu^{k+1} = U_{INW}(x^k, \mu^k, \rho^k),$$

where

$$U_{INW}(x^k, \mu^k, \rho^k) = \mu^k + \Delta\mu.$$

Step 6. Check the validation rule. If

$$T_{nw1} := \max_{i=1,\dots,r} \frac{|\mu_i^{k+1} - \widetilde{\mu}_i|}{|\widetilde{\mu}_i|} \leq \tau_{nw1},$$

(see §2.3) the algorithm finishes with update U_{INW}, achieved in step 5. Otherwise, it ends with $\mu^{k+1} = \widetilde{\mu}$.

2.3. Activation and validation rules

One issue that is interesting to consider is whether it is worth attempting to compute the superlinear-order update when the current point is not close enough to an optimizer. The main reason for not doing so is the high computational cost of obtaining a projected Jacobian J_k having full column rank in the first step of Algorithm 1 (if possible). We run the risk of finding that the current projected Jacobian J_k does not have full column rank, in which case we have wasted all the effort taken to compute it. One way of controlling the closeness to (x^*, μ^*) is given by the following test:

$$T_{nw0} := T_x \text{ and } T_\mu,$$

where, for a previously fixed tolerance τ_x,

$$T_x := \overline{\tau} \leq \tau_x$$

is a test carried out at the end of the solution of subproblem **EGS** (5) in order to know the closeness of the current point to x^*, $\overline{\tau}$ being a tolerance used in the definition of the optimality tolerance for subproblem **EGS** (for further details see the definition of $\overline{\tau}$ in Step 1 of the Algorithm 2 for solving problem **IGP** in §3.1); and

$$T_\mu := ||\mu^k - \mu^{k-1}|| < (1 + ||\mu^k||)\tau_\mu,$$

where τ_μ is a previously fixed tolerance.

In the event of T_{nw0} being false the superlinear-order update is not performed and $\mu^{k+1} = \widetilde{\mu}$, which is the first-order update U_O.

The test

$$T_{nw1} := \max_{i=1,\dots,r} \frac{|\mu_i^{k+1} - \widetilde{\mu}_i|}{|\widetilde{\mu}_i|} \leq \tau_{nw1},$$

is based on the fact that the second-order update is rarely correct if it is very far from the first-order update, see [11]. Therefore, there is some level of agreement between the two estimations.

2.4. A more numerically-stable alternative

In this alternative the second-order update is performed in two steps, as suggested by Bertsekas in proposition 2.8 of [2].

The first step consists of

- computing the first-order estimation of μ^* given by
$$\widetilde{\mu} = U_O(x^k, \mu^k, \rho^k),$$
- taking $\widehat{H}_k = Z_A^t \nabla^2_{xx} L_0(x^k, \widetilde{\mu}) Z_A$, where $L_0(x^k, \widetilde{\mu})$ means the Lagrangian function (i.e., the augmented Lagrangian function for $\rho = 0$) with $\mu = \widetilde{\mu}$,
- and computing a modified Cholesky factorization of $\widehat{H}_k$ (see [12]), obtaining $\widehat{H}_k = \widehat{R}_k^t \widehat{R}_k$.

Note that in Algorithm 1 we replace R_k with $\widehat{R}_k$ and that
$$H_k = \widehat{H}_k + \rho^k J_k J_k^t.$$
In the second step Algorithm 1 is applied replacing the Step 5 with

$$U_{INW}(x^k, \mu^k, \rho^k) = \widetilde{\mu} + \Delta\mu. \tag{23}$$

It is easy to see that the solution obtained with the two procedures is the same, see §2.3.2 in [2].

The fundamental difference between both methods lies in that instead of the reduced Hessian (or an approximation of it) of the augmented Lagrangian function $L_{\rho^k}(x^k, \mu^k)$, we require the reduced Hessian (or an approximation of it) of the Lagrangian function $L_0(x^k, \widetilde{\mu})$, where $\widetilde{\mu}$ also appears instead of μ^k. This precludes using the BFGS approximation of $\nabla^2_{xx} L_{\rho^k}(x^k, \mu^k)$ in each k-th major iteration. However, if we have the analytic reduced Hessian of the Lagrangian function for pair $(x^k, \widetilde{\mu})$ (or an approximation of it, e.g., by finite differences) then the procedure considered in this section can be applied.

Since $\widetilde{\mu}$ is a better estimate than μ^k —using known results for Newton's method— we expect that (23) will yield a vector μ^{k+1} which is closer to μ^* than μ^k.

Moreover an important advantage of this procedure is that although the condition number of
$$J_k^t \widehat{H}_k^{-1} J_k$$

is present in the error bound of the system to solve, here it is multiplied by a $||\Delta\mu||$, which is generally small compared with that obtained directly from step 5 without modifications, and this reduces the effect of the condition number with regard to the direct application of the Algorithm 1 —see §2.7.2 in [13]—, as can be observed in the computational tests later.

2.5. Issues concerning the quasi-Newton multiplier update

A quasi-Newton method (here BFGS) is normally used when the number of superbasic variables is no greater than a number previously given $\bar{s}$, which is chosen so that the computational effort of computing the search direction in this way will be smaller than carrying out this calculation using another efficient method (e.g., the Truncated Newton [8]). Here, $\bar{s} = 500$ by default. If $s > \bar{s}$ it is clear that we will not have this factored approximation, hence we will necessarily either have to compute the reduced Hessian of the augmented Lagrangian directly, or approximate it by means of finite differences according to the procedure explained below. In large-scale problems with many degrees of freedom the possibility of $s > \bar{s}$ is also an issue to take into account at the moment of implementing the quasi-Newton techniques for the superlinear-order update, though it is not a crucial question for the aims of this work. Nevertheless, we must not forget that there are other algorithmic choices that help to solve this problem as for example: the L-BFGS method of Nocedal [25, 28] and the partitioned quasi-Newton method of Griewank and Toint [14].

When using a BFGS approximation in the solution $\tilde{x}$ of subproblem **EGS**, see (5), H will be positive definite and one can directly update its Cholesky factorization. However, the matrix J that we have in $\tilde{x}$ may not have full column rank. Therefore, it will be necessary to take columns of N (defined by (16)) to complete its rank, and the reduction matrix (18) will thus have to be modified as a consequence of enlarging S_A with columns of N_A, which makes the current quasi-Newton update H useless as an approximation of $Z_A^t \nabla_{xx}^2 L_\rho(\tilde{x},\mu) Z_A$ in system (20). Hence the quasi-Newton approximation can only be directly exploited when the reduced Jacobian J at $\tilde{x}$ does not require that columns of N be taken in order to complete the column rank of BS and, in consequence, the rank of J. Otherwise, H could be updated by adding nonbasic variables associated with the columns of N to the superbasic set one by one and updating for each new superbasic variable the Cholesky factorization of H (as in [26]). It is worth bearing in mind when choosing these nonbasic variables that it is appropriate to show preference for those whose multiplier estimate is zero or small enough, in order to avoid the vector Δx derived from the superlinear-order update being infeasible. Let $\widehat{N}_A$ denote the submatrix constituted by these columns of N_A; thus, a new variable reduction matrix $\overline{Z}_A$ is defined by

$$\overline{Z}_A = [Z_A \quad Z_{\widehat{N}}] = \begin{bmatrix} -B_A^{-1}S_A & -B_A^{-1}\widehat{N}_A \\ \mathbb{1} & 0 \\ 0 & \mathbb{1} \\ 0 & 0 \end{bmatrix}. \tag{24}$$

Two rules can be used to exploit the current factorization of the quasi-Newton approximation of the reduced Hessian of the augmented Lagrangian.

1. *Updating of the Cholesky by finite differences.* Denote

   $$\widehat{Z}_A^{(i)} = [Z_A, z_1, \cdots, z_i] = [\widehat{Z}_A^{(i-1)}, z_i],$$

   for $i = 1, \cdots, m_1$, where m_1 is the column number of $\widehat{N}_A$ and z_i denotes the i-th column of $Z_{\widehat{N}}$ such that $\widehat{Z}_A^{(0)} = Z_A$ and $\widehat{Z}_A^{(m_1)} = \overline{Z}_A$.

   The procedure used here consists in repeating successively from $i = 1$ to $i = m_1$ the classical method suggested in [26] for updating the Cholesky factor R when adding a new superbasic variable, in which $\overline{Z}$, Z and z are replaced, for each i, by $\widehat{Z}_A^{(i)}$, $\widehat{Z}_A^{(i-1)}$ and z_i respectively.

2. *Another alternative to keep the positive definiteness.* A new approximation $\widetilde{H}_k$ is taken such that:

   $$\widetilde{H}_k = \begin{bmatrix} H_k & 0 \\ 0 & \mathbb{1} \end{bmatrix},$$

   where each element of $\mathbb{1}$ correspond to a variable whose corresponding column in A belong to $\widehat{N}_A$. Therefore, in order to factor $\widetilde{H}_k = \widetilde{R}_k^t \widetilde{R}_k$ it is enough to set

   $$\widetilde{R}_k = \begin{bmatrix} R_k & 0 \\ 0 & \mathbb{1} \end{bmatrix}.$$

2.6. Approximation of the reduced Hessian of the augmented Lagrangian by finite differences

Next, an approximation of the reduced Hessian of the augmented Lagrangian is calculated by extending the usual lines in the approximation procedure of the Hessian by means of finite differences (see [12]).

Denote $g(x) = \nabla_x L_\rho(x,\mu)$ and $G(x) = \nabla^2_{xx} L_\rho(x,\mu)$. The Taylor-series expansion of g about x allows us to write $g(x+\sigma d) \simeq g(x) + \sigma G(x) d$. Premultiplying by the variable reduction matrix Z_A^t and taking into account that $d = Z_A d_S$, we obtain

$$Z_A^t g(x + \sigma Z_A d_S) \simeq Z_A^t g(x) + \sigma Z_A^t G(x) Z_A d_S,$$

from which

$$Z_A^t G(x) Z_A d_S \simeq \frac{Z_A^t g(x + \sigma Z_A d_S) - Z_A^t g(x)}{\sigma}.$$

Let us consider a given $\sigma > 0$, sufficiently (but not too) small, $d_S = e_i$ is successively taken for $i = 1, \cdots, s$, where e_i is the unit vector that has the one in the i-th position, the remaining entries being zero, and s is the number of columns of matrix Z_A. Then, we can obtain an approximation of the i-th column of $Z_A^t G(x) Z_A$, which is denoted by $\overline{H}^{(i)}$, by means of

$$\overline{H}^{(i)} = \frac{Z_A^t g(x + \sigma Z_A e_i) - Z_A^t g(x)}{\sigma},$$

thus achieving $\overline{H} \simeq Z_A^t G(x) Z_A$.

Once $\overline{H}$ is computed a symmetrical approximation H is obtained by doing $H_{ij} =$ $= 1/2(\overline{H}_{ij} + \overline{H}_{ji})$ in order to calculate each entry (i, j) of H; in passing, $\overline{H}_{ij}$ and $\overline{H}_{ji}$ are compared, and if the difference between them is excessive, this approximation must be rejected and a more suitable σ must be chosen. One classic possibility is to take $\sigma = \sqrt{\varepsilon_M}$ (ε_M=machine precision).

3. EXTENSION OF THE NEWTON-LIKE METHODS TO PROBLEM IGP

In [21] the original multiplier method is extended to problems of the **IGP** type (see (7–10) in §1), specializing its performance in the case of nonlinear network flows with side constraints. There an algorithm for solving this problem using the aforementioned method is put forward, which is summarized in §3.1.

When we have $\alpha = \beta = 0$, problem **IGP** becomes problem **EGP**. As can be seen in §2, in the solution of this problem, subproblem **EGS** (5) must be solved and the vector μ updated.

Here the extension of update U_{INW} of §2.1 to problem **IGP** for $\alpha < \beta$ is considered. At each iteration of the multiplier method we solve subproblem **(IGS)**:

$$\begin{array}{lll} & \underset{x}{\text{minimize}} & L_\rho(x, \mu) \\ (25) & \text{subject to} & Ax = b \\ & & l \le x \le u, \end{array}$$

where $\rho > 0$ and μ are fixed,

$$(26) \qquad L_\rho(x, \mu) = f(x) + \sum_{j=1}^{r} p_j[c_j(x), \mu_j, \rho],$$

p_j being defined by

$$p_j[c_j(x),\mu_j,\rho] = \mu_j\varphi_j[c_j(x),\mu_j,\rho] + \frac{1}{2}\rho|\varphi_j[c_j(x),\mu_j,\rho]|^2, \tag{27}$$

such that

$$\varphi_j[c_j(x),\mu_j,\rho] = \begin{cases} c_j(x)-\beta_j & \text{if } \mu_j+\rho[c_j(x)-\beta_j]>0 \\ c_j(x)-\alpha_j & \text{if } \mu_j+\rho[c_j(x)-\alpha_j]<0 \\ -\mu_j/\rho & \text{otherwise.} \end{cases} \tag{28}$$

Replacing in (27) $\varphi_j[c_j(x),\mu_j,\rho]$ with its expression in (28) and operating we get the function $p(t,\mu,\rho)$ defined by Bertsekas in §3.2 of [2], which is continuously differentiable with respect to t and twice continuously differentiable for all t except $t=\alpha-\mu/\rho$ and $t=\beta-\mu/\rho$, see [2].

In [21] the following proposition is shown.

Proposition 1 (a) *If f and c are continuous in a subset $\mathcal{X}$ of $\mathbb{R}^n$, $L_\rho(\cdot,\mu)$ is continuous in $\mathcal{X}$ for all μ, and* $\rho>0$.

(b) *If f and c are continuously differentiable in an open subset $\mathcal{X}$ of $\mathbb{R}^n$, $L_\rho(\cdot,\mu)$ is continuously differentiable in $\mathcal{X}$ for all μ, and $\rho>0$.*

(c) *Let $K=\{1,\dots,r\}$. If f and c are twice continuously differentiable in an open subset $\mathcal{X}$ of $\mathbb{R}^n$, $L_\rho(\cdot,\mu)$ is twice continuously differentiable in the set*

$$\begin{aligned} \widehat{\mathcal{X}}_{\mu,\rho} = \quad & \{x \mid x\in\mathcal{X},\ \mu_j+\rho[c_j(x)-\beta_j]\neq 0,\ j\in K\} \\ & \cap\{x \mid x\in\mathcal{X},\ \mu_j+\rho[c_j(x)-\alpha_j]\neq 0,\ j\in K\} \end{aligned} \tag{29}$$

for all μ, and $\rho>0$.

On the other hand, let E denote the *state vector* defined for a given pair (μ,ρ) as follows:

$$E_j = \begin{cases} +1, & \text{if } c_j[x(\mu,\rho)]-\beta_j > -\mu_j/\rho \\ -1, & \text{if } c_j[x(\mu,\rho)]-\alpha_j < -\mu_j/\rho \\ 0, & \text{otherwise.} \end{cases} \tag{30}$$

Let $\widehat{c}(x)$ denote the active contraints of $c(x)$ (we say that $c_j(x)$ is active in the sense that $E_j\neq 0$; see (30)) and $\widehat{\mu}$ its corresponding multiplier vector. Moreover, the zero superscript (e.g., as in $\Delta\mu^0$ and μ^0/ρ) means that the symbol with which is associated refers to the inactive constraints (in the sense that $E_j=0$), and $\widehat{\varphi}$ stands for the current value of the vector function constituted by the φ_j functions (defined in (28)) corresponding to those constraints $c_j(x)$ which are active. Using this notation, let us consider any update formula U of μ in the solution of **IGP** by means of multiplier methods, where the subproblem to solve is **IGS** (25). In **IGP** the first-order multiplier estimate is given as

$$U_0(x,\mu_j,\rho) = \begin{cases} \mu_j + \rho\varphi_j[c_j(x),\mu,\rho], & \text{if } E_j \neq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{31}$$

Let us assume that $(x^*,\mu^*,\pi^*,\lambda^*)$ is an optimizer at the end of the sequential procedure satisfying the strict complementary condition, then there exists a neighborhood $N(\mu^*)$ such that for $\mu \in N(\mu^*)$, vector $x(\mu,\rho)$ belongs to $\widehat{\mathfrak{X}}_{\mu,\rho}$, see (29).

First-order optimality conditions to be fulfilled by the optimizer $(x^*,\mu^*,\pi^*,\lambda^*)$ for all $\rho > 0$ large enough, are —in addition to the complementary slackness condition over λ^* (strict by hypothesis)— system

$$\begin{array}{rcl} \nabla_x L_\rho(x,\mu) + A^t\pi + \lambda & = & 0 \\ \varphi[c(x),\mu,\rho] & = & 0 \\ Ax & = & b, \end{array} \tag{32}$$

where

$$\nabla_x L_\rho(x,\mu) = \nabla f(x) + \nabla c(x)\Big(\mu + \rho\varphi[c(x),\mu,\rho]\Big).$$

Let $\widetilde{x} = x(\mu,\rho)$ be a solution of subproblem **IGS**, see (25). Bearing in mind that both φ and $\nabla_x L_\rho(x,\mu)$ are differentiable with respect to x in $\widehat{\mathfrak{X}}_{\mu,\rho}$, the solution of system (32) by means of Newton's method at $x = \widetilde{x}$ means solving the system

$$\begin{array}{|c|c|c|c|c|} \multicolumn{1}{c}{n} & \multicolumn{1}{c}{r_0} & \multicolumn{1}{c}{\widehat{r}} & \multicolumn{1}{c}{m} & \multicolumn{1}{c}{\widehat{t}} \\ \hline \nabla^2_{xx}L_\rho(\widetilde{x},\mu) & 0 & \nabla\widehat{c}(\widetilde{x}) & A^t & \begin{array}{c} 0 \\ \hline \mathbb{1} \end{array} \\ \hline 0 & \frac{-1}{\rho} & & & \\ \cline{1-2} \nabla\widehat{c}(\widetilde{x})^t & & & 0 & \\ \cline{1-1} A & & & & \\ \cline{1-1} \begin{array}{c|c} 0 & \mathbb{1} \end{array} & & & & \\ \hline \end{array} \quad \begin{array}{|c|} \hline \Delta x \\ \hline \Delta\mu^0 \\ \hline \Delta\widehat{\mu} \\ \hline \Delta\pi \\ \hline \Delta\lambda \\ \hline \end{array} = \begin{array}{|c|} \hline -\nabla_x L_\rho(\widetilde{x},\mu) \\ -A^t\pi - \lambda \\ \hline \mu^0/\rho \\ \hline -\widehat{\varphi} \\ \hline 0 \\ \hline 0 \\ \hline \end{array}, \tag{33}$$

where $\frac{-1}{\rho}$ stands for the diagonal matrix whose nonzero entries are equal to $-1/\rho$. Note that necessarily $n > m + \widehat{r} + \widehat{t}$ and $r_0 + \widehat{r} = r$.

As in the case of system (14), we can premultiply system (33) with matrix $\overline{Z}$ defined as in (17), consider that the matrix of coefficients is nonsingular and use the equality (19), in order to finally obtain the system

$$\begin{array}{ccc} s & r_0 & \hat{r} \\ \hline \multicolumn{1}{|c|}{Z_A^t \nabla^2_{xx} L_\rho(\tilde{x},\mu) Z_A} & \multicolumn{1}{c|}{0} & \multicolumn{1}{c|}{Z_A^t \nabla \hat{c}(\tilde{x})} \\ \hline \multicolumn{1}{|c|}{0} & \multicolumn{1}{c|}{\frac{-1}{\rho}} & \multicolumn{1}{c|}{} \\ \cline{1-2} \multicolumn{1}{|c|}{\nabla \hat{c}(\tilde{x})^t Z_A} & & \multicolumn{1}{c|}{0} \\ \hline \end{array} \begin{array}{|c|} \hline \Delta x_S \\ \hline \Delta \mu^0 \\ \hline \Delta \hat{\mu} \\ \hline \end{array} = \begin{array}{|c|} \hline -Z_A^t \nabla_x L_\rho(\tilde{x},\mu) \\ \hline \mu^0/\rho \\ \hline -\hat{\varphi} \\ \hline \end{array}, \tag{34}$$

which is called the *reduced Newton system associated with problem* **IGP**. The solution of this system provides $(\Delta x_S, \Delta\mu^0, \Delta\hat{\mu})$.

Note that subsystem associated with submatrix $-\mathbb{1}/\rho$ is fully independent of the rest, its solution being

$$\Delta\mu^0 = -\mu^0,$$

where μ^0 is the value of the multiplier vector of the inactive constraints before it is updated, from which we find that the multiplier estimate of these constraints after being updated is zero. Therefore, the system to solve is the same as (20), but replacing $\Delta\mu$, $\nabla c(\tilde{x})$, and $c(\tilde{x})$ with $\Delta\hat{\mu}$, $\nabla\hat{c}(\tilde{x})$, and $\hat{\varphi}$ respectively. The solution is performed by means of the Algorithm 1, but with the above substitutions. Therefore for problem **IGP** we have

$$U_{INW}(x,\mu_j,\rho) = \begin{cases} \mu_j + \Delta\hat{\mu}_j & \text{if } E_j \neq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{35}$$

3.1. Algorithm for solving problem IGP

The techniques for the partial elimination of constraints will be used so as to relax only the general constraints (9) of problem **IGP** (see (7–10)), i.e. both constraints (8) and (10) being explicitly maintained. In this way, for each pair (μ^k, ρ^k), the following partial augmented Lagrangian is obtained:

$$L_{\rho^k}(x,\mu^k) = f(x) + \sum_{j=1}^{r} p_j[c_j(x), \mu_j^k, \rho^k], \tag{36}$$

where the function p_j is defined as in (27–28). This gives rise to the subproblem **($\mathbf{IGS_k}$)** —see the definition of **IGS** in (25)—, which will be solved approximately for each k *major iteration* by Murtagh and Saunders's active set technique. The minimization procedure for $\mathbf{IGS_k}$ will be interrupted when vector $x^k = x(\mu^k, \rho^k)$ satisfies a predetermined stopping criterion which becomes more rigorous as k increases, so that the minimization will be *asymptotically exact*, as indicated in §2.5 of [2]. In the development of the algorithm, for each k major iteration, the *state vector* E^k (see (30)) and the *violation vector* V^k are used, where

$$V_j^k = \begin{cases} c_j(x^k) - \beta_j, & \text{if } E_j^k = +1 \\ c_j(x^k) - \alpha_j, & \text{if } E_j^k = -1 \\ 0, & \text{otherwise.} \end{cases} \tag{37}$$

Algorithm 2 (Multiplier method associated with IGP)

Step 0. Initialization. μ^0, ρ^0, γ_0 and the auxiliary vector V^{-1} are appropriately selected; set $k = 0$ and tolerances ε_{opt} and τ are set to sufficiently small values.

Step 1. Solution of subproblem. Subproblem $\mathbf{IGS_k}$ is solved by Murtagh and Saunders's active set technique. The subproblem is considered solved when the algorithm associated with that active set technique stops for an optimality tolerance equal to τ_k, which is defined as

$$\tau_k = \overline{\tau} * \nu(\pi_k),$$

where

$$\overline{\tau} = \max\{\overline{\varepsilon}_k, \varepsilon_{opt}\}, \quad \overline{\varepsilon}_k = \min\{\varepsilon_k, \gamma_k \|V^k\|\}, \text{ and}$$

$$\nu(\pi_k) = \max\left\{1, \frac{\|\pi_k\|_1}{\sqrt{m}}\right\},$$

π_k being the multiplier vector associated to the constraints $Ax = b$ at the local minimum obtained for $\mathbf{IGS_k}$, x^k. ($\nu(\pi)$ is a function used by MINOS 5.5 [27].)

Step 2. Feasibility of x^k with respect to the general constraints. If $T_{opt} := \overline{\tau} \leq \varepsilon_{opt}$,

(a) if at x^k,

$$\|V^k\|_\infty < \tau \|c(x^k)\|,$$

is fulfilled, go to step 7,

(b) otherwise, go to step 3.

If T_{opt} is not true, ε_k and γ_k are updated so that $\{\varepsilon_k\} \longrightarrow 0$ and $\{\gamma_k\} \longrightarrow 0$ by means of

$$\varepsilon_{k+1} = \theta_1 \varepsilon_k$$

$$\gamma_{k+1} = \theta_2 \gamma_k,$$

such that $0 < \theta_i < 1$ for $i = 1, 2$. Go to step 3.

Step 3. Multiplier update. The vector μ that calculates the general constraint multipliers is updated following the expresion

$$\mu_j^{k+1} = U(x^k, \mu_j^k, \rho^k),$$

where U means any update formula of μ.

Step 4. Update of penalty parameter. The penalty parameter ρ^k is updated if the violation of any active constraint j has not been sufficiently reduced. That is, if there exists any j, such that $E_j^k \neq 0$, for which it is verified that $|V_j^k| > \frac{1}{\gamma}|V_j^{k-1}|$, then set

$$\rho^{k+1} = \xi \rho^k,$$

where ξ and $\overline{\gamma}$ are such that $\xi, \overline{\gamma} \in (1, 10]$ with $\xi > \overline{\gamma}$.

Step 5. Set $k = k + 1$.

Step 6. Return to step 1 with x^{k-1} as the initial feasible point for the subproblem $\mathbf{IGS_k}$.

Step 7. Optimum solution found. The algorithm is stopped: a local minimum of problem **IGP** has been found.

The implementation of this algorithm when it is specialized for networks gives rise to the code PFNRN. More information about this algorithm and its implementation (for $U = U_0$ in Step 3, see (31)) can be found in [21].

4. IMPLEMENTATION

The algorithm that is implemented here has Algorithm 1 as an alternative Step 3 in Algorithm 2. The implementation in Fortran-77, called PFNRN03, was designed mainly to solve large-scale **IGP** type problems (see (7–10)) when these are nonlinear network problems with nonlinear side constraints, given the existence of efficient algorithms to solve this kind of problem when the side constraints are linear. It makes use of the network structure to improve its efficiency, taking into account the specific procedures for nonlinear networks flows [9, 31] and Murtagh and Saunders's active set procedure using a spanning tree as the basis matrix of the network constraints. PFNRN03 also exploits the sparsity of the Jacobian and of the Hessian (as defined by the user).

In this code, the user can set up the level of accuracy of the optimizer by means of the parameter ε_{opt}, whose default value is 10^{-6}. Note that $\{\overline{\tau}\} \longrightarrow \varepsilon_{opt}$ as $\{\varepsilon_k\} \longrightarrow 0$ —i.e., an asymptotically exact minimization is used, as Bertsekas suggests in [2].

For Step 2 the user can also set up a level of feasibility accuracy for the side constraints by giving a value to the parameter τ. This value is 10^{-5} by default. Also, the values of ε_0 and γ_0 can be modified, $\varepsilon_0 = 0.01$ and $\gamma_0 = 0.25$ being the default ($\theta_1 = 0.5, \theta_2 = 0.25$ are fixed).

In Step 3, when it is performed by Algorithm 1, the values of the tolerances in the activation and validation rules are $\tau_x = \varepsilon_{opt}$, $\tau_\mu = 0.1$, and $\tau_{nw1} = 1.0$.

The last values have been chosen in order to have a sufficiently large amount of major iterations where the superlinear-order estimation is used, it allows us to evaluate the performance of this estimator. The initial multiplier estimate vector μ^0 is the null vector by default, although it is worth using information on this vector when it is available, given the substantial improvement that it may bring to bear on the convergence of the algorithm, as was found by means of numerical experiments.

In Step 4 the default values are $\rho^0 = 0.1$ and $\xi = 2.0$. These values can also be modified by the user. Likewise, $\overline{\gamma} = 0.9\xi$, as Bertsekas suggests in [2]. Note that if $|V_j^k| \leq \frac{1}{\gamma}|V_j^{k-1}|$, ρ does not grow in the k-th iteration. Moreover, when the current μ^k is sufficiently close to μ^* —i.e., $||\mu^{k+1} - \mu^k||_1 < \eta_0(1 + ||\mu^k||_1)$ for a small enough scalar η_0 (here $\eta_0 = 0.01$)— $\xi = 1$, so ρ^k stops growing in order not to increase the ill-conditioning at the end of the execution. In practice this rule is very stringent, hence ρ is sufficiently large when it is true. This technique is used by MINOS 5.5 to reduce ρ or to set it to zero, see [27].

Furthermore, by default, the constraints are initially multiplied by scale factors that make the norm of $\nabla c_j(x^0)$ equal to 1, for $j = 1, \ldots, r$. The user may eliminate this scaling.

This implementation includes the two following choices with respect to Algorithm 1:

1. using this algorithm without changes, giving rise to the update U_{S1}
2. using the algorithmic alternative given in §2.4, giving rise to the update U_{S2}.

In addition, with regard to the reduced Hessian H_k of the augmented Lagrangian function at x^k three different choices have been considered:

(H) the exact reduced Hessian, if it is positive definite, or otherwise the positive definite approximation achieved by means of the modified Cholesky method (see §4.4.2.2 in [12]);

(D) an approximation by finite differences achieved from projected gradients as is put forward in §2.6; and

(Q) a quasi-Newton approach by means of the BFGS update, using the first rule given in §2.5 when the projection J_k of the Jacobian over the manifold defined by the current nonbasic variables does not have full column rank. We must not forget that BFGS update is only used when the number of superbasic variables s is not greater than $\bar{s}$ ($\bar{s} = 500$ by default). In this implementation the alternative method to the BFGS is the Truncated Newton [8]), see §2.5.

In consequence, by combining the two type of updates (U_{S1} and U_{S2}) and the above three choices for the projected Hessian we have several updates that are different in practice, and which are denoted by expressions such as U_{S1H}, U_{S1D} and U_{S1Q}, where these mean that update U_{S1} is calculated by using the alternatives H, D and Q respectively. The same is true for update U_{S2}. If $s > \bar{s}$ when using U_{S1Q}, the alternative update is U_O. Nevertheless, for U_{S2} it is not possible to compute U_{S2Q} because we do not have a quasi-Newton approximation of the reduced Hessian of the Lagrangian function (whereas we do have one of the augmented Lagrangian function).

In this package there are other parameters that can also be modified by means of a specification file, see [20].

The code PFNRN03 and its user's guide [20] can be accessed *for academic purposes* via *anonymous ftp* at `ftp-eio.upc.es` in directory: `pub/onl/codes /pfnrn`. (Comments, suggestions, or description of any trouble or abnormal situations experienced when using it reported to `mepmifee@lg.ehu.es` will be appreciated.)

Code PFNRN03 contains the solver PFNL for nonlinear network flow problems, which is used in Step 1 (subproblem solution) of the Algorithm 2. Any other nonlinear network flow code could have been employed instead.

5. COMPUTATIONAL TESTS

In order to evaluate the efficiency and robustness of code PFNRN03 a series of computational tests are performed, which consist of solving nonlinear network flow problems with linear and nonlinear side constraints with this code in its various versions (depending on the kind of U_S) and comparing it with the results obtained when the multiplier estimation is carried out by means of U_O. All these tests were performed on a Sun Sparc 10/41 work station under UNIX.

Two types of problems are considered: real and artificial. The real problems solved are of Short-Term Hydrothermal Coordination of Electricity Generation [21]. They correspond to problems whose name starts with «P», and they have *two-sided inequality* *nonlinear side constraints*. The objective function in strongly nonlinear, with a high computational cost.

Table 1. Test problems.

test	arcs	nodes	s.c.	j.e.n.	a.s.c	sb.a.
D12e2	1524	360	180	183	3	75
D13e2	1524	360	360	364	9	89
D14e2	1524	360	36	538	31	151
D15e2	1524	360	36	5387	4	107
D16e2	1524	360	36	96	20	83
D13n1	1524	360	360	364	38	681
D21e2	5420	1200	120	135	2	30
D22e2	5420	1200	600	656	13	46
D23e2	5420	1200	120	135	17	238
D31e1	4008	501	5	20	1	63
D32e1	4008	501	50	199	5	83
D31e2	4008	501	5	20	1	60
D32e2	4008	501	50	199	5	65
D41e2	12008	1501	15	182	9	172
D51e1	18000	3000	30	526	8	58
D52e1	18000	3000	150	27118	16	166
P4103	3591	1072	63	3071	10	17
P4101	3591	1072	63	3071	32	63
P4203	5187	1548	91	4443	10	17
P4201	5187	1548	91	4443	32	65
P4403	7581	2262	133	6501	10	20
L13e2	1524	360	360	363	7	67
L21e2	5420	1200	120	138	5	27
L32e2	4008	501	120	138	4	61
L52e1	18000	3000	30	527	10	57
XA48	2256	697	240	1915	145	265

The model that gives rise to the problem «xa» is presented in [16] as a single network model for Hydrothermal Scheduling obtained by joining the hydrogeneration optimization network and a series of thermal subnetworks (one for each interval), all of them with a common sink node. The objective function to be minimized is quadratic with both *equality and inequality linear side constraints*.

Problems created from DIMACS random network generators [10] are used as artificial problems: generators Rmfgen, Gridgen and Grid-on-Torus have been employed, linear side constraints being created from the *Di2no* random generator [17], and they give rise to the problems whose first letter is an «L». The nonlinear side constraints for the DIMACS networks are generated through the *Dirnl* random generator described in [21], and they give rise to whose first letter is an «D». Several types of objective function used to create problems with DIMACS networks. The last two letters in the problem names starting with L or D indicate the type of objective function: «n1» functions are of the Namur family used in [31] and «e1» and «e2» functions belong to the EI01 family, which was designed by Heredia in [17] (see also [21]) with a view to obtaining problems with a moderate number of superbasic variables at the optimizer.

Table 1 shows the characteristics of the problems resulting from these models. The names of the test problems are given in the *test* column. The *arcs* and *nodes* columns give us, respectively, the number of arcs and nodes in the network, whilst *s.c.* and *j.e.n.* give, respectively, the total number of the problem's side constraints and the number of non-null entries in their Jacobian. Finally, the *a.s.c.* and *sb.a.* give, respectively, the number of side constraints that are active and the number of variables that are superbasic at the optimum (superbasic variables with respect to the nonlinear network subproblem that is solved in each major iteration of Algorithm 2).

Next there are two subsections dedicated to the computational results. In the first the results for the various versions of U_S are compared with each other and with U_O. The second part provides only the efficiency evaluation of the quasi-Newton update U_{S1Q} with respect to the first-order update U_O, though for a greater number of test problems than in the first part.

5.1. Comparison of results

Each test problem was solved by using the five types of update U_S considered here. The only aspect that is modified in each solution of any test problem is the type of update U_S with which it is solved. The results have been grouped into two different tables, 2 and 3, as the multiplier update will be U_{S1} or U_{S2}. Each table presents the results for the three types of H_k (projected Hessian matrix of the augmented Lagrangian function, or an approximation of it, in the k-th major iteration) that are used in this implementation —i.e., H, D and Q, see §4— except for U_{S2Q}, given that the quasi-Newton approximation of the projected Hessian of the Lagrangian function is not available. The place of this latter type of update in this table is occupied by the update U_O, which is used as a reference.

In each table, the name of the test problem is given under the heading «TEST»; the columns «Mit», «mit» and «cpu» are used to give, respectively, the major iteration number, the minor iteration number (total number of iterations used to solve the consecutive network subproblems **IGS** (25); see §3.1) and the CPU seconds spent by the execution.

Table 2. Results for update U_{S1} and various H_k.

TEST	U_{S1H}			U_{S1D}			U_{S1Q}		
	Mit	mit	cpu	Mit	mit	cpu	Mit	mit	cpu
D12e2	18	890	27.4	18	890	21.0	14	790	14.4
D13e2	15	2028	45.7	15	2018	45.9	13	1938	30.9
D13n1	14	3652	5582.8	14	3660	750.5			(†)
D23e2	15	3124	344.9	15	3121	231.9	13	3194	161.7
D32e1	35	3802	265.8	35	3802	201.2	9	3951	116.5
D51e1	130	1277	6981.8	121	1285	594.9	8	1116	91.7
P4203			($\star$)	12	4038	428.5	10	3167	328.1
P4403	8	3934	722.2	12	5563	1022.4	9	4780	804.0
L13e2			(‡)			(‡)	9	508	6.1
L21e2	42	1801	119.6	42	1801	76.9	10	1724	22.6
L32e2			(‡)			(‡)	7	856	20.1
L52e1	6	4096	250.5	6	4096	222.5	6	4117	221.8

Table 3. Results for update U_{S2} and various H_k, and U_O.

TEST	U_O			U_{S2H}			U_{S2D}		
	Mit	mit	cpu	Mit	mit	cpu	Mit	mit	cpu
D12e2	25	1184	15.6	18	869	22.5	18	869	17.6
D13e2	20	2497	36.3	15	2064	45.7	15	2052	38.3
D13n1	18	3869	478.3	14	3662	5581.7	14	3649	792.8
D23e2	33	4948	279.0	14	3103	306.6	14	3103	194.9
D32e1	28	5114	136.8	35	4437	280.0	35	4437	218.2
D51e1	12	1170	97.0	8	1099	202.2	8	1099	93.5
P4203	7	3071	303.0	6	2859	277.7	7	2895	291.3
P4403	7	3681	547.4	9	4331	716.3	8	4560	748.4
L13e2	16	537	6.4	8	506	6.7	8	506	5.7
L21e2	39	1848	24.2	14	1963	53.7	14	1963	40.4
L32e2	9	868	21.5	6	849	25.8	6	849	23.2
L52e1	6	4116	219.6	6	4116	252.1	6	4116	221.3

Table 2 shows the results for update U_{S1} (which is directly derived from the Algorithm 1) with the three choices of the matrix H_k considered in §4. Note that the running times of U_{S1H} are greater, in nearly all cases, than those of U_{S1D}, at the same time as the running times of U_{S1D} are greater than those of U_{S1Q} in all cases. The reason of this lies in the definition of H_k and in the way in which this matrix is computed. If H is the authentic reduced Hessian of the augmented Lagrangian one must compute the Hessian of the augmented Lagrangian —$\nabla^2_{xx}L_\rho(x,\mu)$— and then its projection —$Z^t_A\nabla^2_{xx}L_\rho(x,\mu)Z_A$—, whereas if H is an approximation by finite differences achieved from projected gradients as is put forward in §2.6, H is directly computed column by column. On the other hand, if H is a quasi-Newton approximation (see §4), it is supplied together with the solution of the subproblem.

In this table (†) means that problem «D13n1» cannot be solved by using a quasi-Newton approach because in this problem the number of superbasic variables is almost 700, higher than $\overline{s}$ (here 500), and when this happens the method used is the Truncated Newton (see §4), which leads to this approximation being unavailable in problem «D13n1». In addition, (‡) means that the execution stops at the end of a large number of major iterations (e.g., in «L13e2» more than 480) since it does not succeed in improving the current point, as a consequence of a bad updating of the multipliers, which can be attributed to numerical stability difficulties in the solution of Step 4 in Algorithm 1. However, this trouble does not appear in the solution of the same problem by U_{S2}, as can be seen in Table 3, which confirms the greater numerical stability of the update U_{S2}, see the definition of U_{S2} in §4. In «P4203» $(\star)$ means that the execution stops at the end of 14 major iterations and 4533 minor iterations as a result of the bad convergence caused by an inaccurate estimation of the multipliers. This in turn is a consequence of the reasons given for (‡), together with the fact that, when using the modified Cholesky factorization, the Frobenius norm of the modification of the projected Hessian used to make it into positive definite is of the order of 10^{-3}. However, Table 3 shows how the results for this problem improve when U_{S2} is used. In Table 3 the symbols (†), (‡) and $(\star)$ (if they appear) mean the same as in Table 2.

As is shown in Tables 2 and 3, and as is expected in theory, if we compare for each problem the number of minor iterations when U_S is used, it is generally lower than when U_O is used, whereas, in contrast, the running times are higher. This last happens because of the high computational cost of calculating the superlinear-order estimate of the multiplier vector with regard to the low cost of using U_O. Moreover, it could happen that in Algorithm 1 condition T_{nw0} would hold, but T_{nw1} would not; then all the computations carried out in this algorithm would have been wasted.

On the other hand, the estimation of the multipliers in problems «P4203» and «P4403» is not very good because of the two following reasons: the projected Hessian of the augmented Lagrangian function is indefinite in these problems and the aforementioned issues of numerical stability (e.g., in «P4403» the Frobenius norm of the modification matrix of the modified Cholesky factorization has occasional values of the order of 10^{7}).

The high number of major iterations in problem «D51e1» when it is solved by using U_{S1H} and U_{S1D} is due to the active side constraints having —in some cases— a very low activity, and matrix H_k being indefinite but almost positive semidefinite. Hence the modification matrix of the Cholesky factorization has a Frobenius norm higher than zero, which yields numerical stability problems. However, these difficulties are not encountered when using U_{S2H} and U_{S2D} due to the reasons given above.

An important effect noticed in many of these tests, and which occurs when using any U_S superlinear-order update, is that although the initial conditions are the same, the final value of the penalty parameter required for the convergence of the algorithm can be much smaller than that obtained when using the first-order update; e.g., in problem «D23e2», $\rho^* = 33554432$ for U_O and, however, $\rho^* = 40$ for any U_S, the initial values being always $\rho^0 = 10$ and $\xi = 2$, see Algorithm 2 in §3.1. Hence, as a consequence of performing the multiplier estimation with U_S updates, the ill-conditioning is reduced when solving subproblem **IGS** (25) in each major iteration. This effect has two numerical consequences: firstly, the algorithm converges faster when a U_S update is used, reducing the number of minor iterations (as is shown in these tables); and secondly, the execution with U_O of some tests breaks down because of ill-conditioning, whereas with U_S it finished succesfully.

Table 4. Comparison of efficiency with regard to U_O.

TEST	U_{S1H}	U_{S1D}	U_{S1Q}	U_{S2H}	U_{S2D}
D12e2	0.57	0.74	**1.08**	0.69	0.89
D13e2	0.79	0.79	**1.17**	0.79	0.95
D13n1	0.09	0.64	(†)	0.09	0.60
D23e2	0.81	**1.20**	**1.73**	0.91	**1.43**
D32e1	0.51	0.68	**1.17**	0.49	0.63
D51e1	0.01	0.16	**1.06**	0.48	**1.04**
P4203	$(\star)$	0.71	0.92	**1.09**	**1.04**
P4403	0.76	0.54	0.68	0.76	0.73
L13e2	(‡)	(‡)	**1.05**	0.96	**1.12**
L21e2	0.20	0.31	**1.07**	0.45	0.60
L32e2	(‡)	(‡)	**1.07**	0.83	0.93
L52e1	0.88	0.97	0.99	0.87	0.99

In order to give a clearer idea of the efficiency of the different U_S updates in comparison with the update U_O, Table 4 has been constructed (where the symbols (†), (‡) and $(\star)$ mean the same as in Table 2). For this same reason we have used bold type for all those efficiency values that are one or greater than one, where each efficiency value (*ev*) is given by the following ratio:

$$ev(XY) = \frac{\text{CPU time using } U_O}{\text{CPU time using } U_{SXY}}, \tag{38}$$

X being either 1 or 2, and Y being H, D or Q, excluding $XY = 2Q$ for the reasons given at the beginning of this section. Therefore, $ev(XY) \geq 1.00$ means that the efficiency of U_{SXY} is at least as good as that of U_O, and hence, the greater $ev(XY)$, the greater the efficiency of U_{SXY} with respect to U_O. Below each heading U_{SXY} we find the different values of $ev(XY)$ for each test problem.

Since the interesting results for the update U_{S1Q} at Table 4, we compare in a more detailed way the efficiency of this update with respect to the first-order update U_O in the following section, in order to decide whether it is worth using U_{S1Q} instead of U_O.

5.2. Efficiency of the update U_{S1Q} regarding U_O

Here we consider the results of Table 5, where the headings are the same as those used in previous tables. The symbol (†) means that the execution was stopped due to ill-conditioning, and (‡) means that since it is not possible to obtain a projected Jacobian having full rank in any major iteration, update U_{S1Q} offers the same results as update U_O. The reason that this Jacobian matrix cannot be found is that some of the active constraints have normals that are orthogonal to the subspace spanned by the columns of matrix Z_A —see (18)— in each major iteration.

Table 5. Efficiency of U_{S1Q} with regard to U_O.

TEST	U_O			U_{S1Q}			$ev(1Q)$
	Mit	mit	cpu	Mit	mit	cpu	
D12e2	25	1184	15.6	14	790	14.4	1.08
D13e2	20	2497	36.3	13	1938	30.9	1.17
D14e2	26	5485	147.5	20	4006	117.1	1.26
D15e2			(†)	19	1909	56.87	(†)
D16e2	850	3966	85.4	16	2345	39.0	2.19
D21e2	22	48669	1155.1	17	2238	40.1	28.81
D22e2	19	5306	119.5			(‡)	(‡)
D23e2	33	4948	279.0	13	3194	161.7	1.73
D31e1	10	1253	29.1	7	1085	30.9	0.94
D32e1	28	5114	136.8	9	3951	116.5	1.17
D31e2	9	947	24.9	7	535	20.0	1.25
D32e2			(†)	17	3661	110.9	(†)
D41e2	32	2842	301.5	10	2143	265.2	1.14
D51e1	12	1170	97.0	8	1116	91.7	1.06
D52e1	10	8388	1258.5	11	8511	1362.9	0.92
P4103	10	2814	165.3	12	2852	177.6	0.93
P4101	8	23687	1520.3	10	21929	1355.3	1.12
P4203	7	3071	303.0	10	3167	328.1	0.92
P4201	7	22845	2159.7	8	22901	2161.1	1.00
P4403	7	3681	547.4	9	4780	804.0	0.68
L13e2	16	537	6.4	9	508	6.1	1.05
L21e2	39	1848	24.2	10	1724	22.6	1.07
L32e2	9	868	21.5	7	856	20.1	1.07
L52e1	6	4116	219.6	6	4117	221.8	0.99
XA48	175	1465	103.8	27	1296	89.5	1.16

Note that except for the problems «D22e2» and «P4403», the efficiency of update U_{S1Q} is similar to or higher than that of U_O, reaching an efficiency value of 28.81 for problem «D21e2» and a value of 2.19 for problem «D16e2». The mean value of the computed efficiencies excluding the highest and lowest values is 1.16. This is not very much larger than one, but we must consider it together with the higher robustness of U_{S1Q}. As an example of this, see the problems «D15e2» and «D32e2» in Table 5, in both cases the execution was stopped due to ill-conditioning. In fact, for «D32e2» when this stopped the penalty parameter value was 65538, whereas using U_{S1Q} the execution normally finished with $\rho_S^* = 4$. Therefore, in the face of the classical difficulty of the augmented Lagrangian techniques the U_S updates are a good alternative, in particular it is worth taking update U_{S1Q} (rather than the first-order update U_O) seriously into account when using augmented Lagrangian techniques with large scale problems, specially with nonlinear networks with nonlinear side constraints.

6. CONCLUSIONS

In this paper the author has put forward techniques for implementing superlinear-order multiplier estimations using the least computational effort.

The projected Jacobian may not have full rank —although it will at the optimum— if the prediction of the active variables at the optimizer provided in the solution of the augmented Lagrangian subproblem is not totally correct, because of the fact that the current point is not sufficiently close to the optimum. Therefore, caution should be exercised when computing the multiplier estimation either to ensure proximity to the optimum or otherwise to avoid using a Newton method, as in this case we do not have the help of the line search.

An algorithm is designed that allows us to solve nonlinear problems with linear constraints, simple bounds, and two-sided nonlinear constraints, by combining superlinear and first-order multiplier methods together with variable reduction techniques. This procedure is particularly interesting in case that the linear constraints are flow conservation equations, as there exist efficient techniques to solve nonlinear network problems.

An implementation of the designed algorithm is put forward, which is specialized for network problems and offers various choices for computing a superlinear-order multiplier estimation. As a result the code PFNRN03 is obtained.

The computational results confirm the quasi-Newton update as a better and more robust alternative than the first-order multiplier estimation to solve large-scale nonlinear network problems with linear or nonlinear side constraints. In addition, we must take into account that since the pure network constraints are the only constraints that are explicitly maintained, whether or not the subproblem solved iteratively is a pure network subproblem only has a significant effect on the total CPU time, but not on the comparison of the efficiency of the different kinds of superlinear-order multiplier estimation used. Furthermore, from the computational tests it is also clear that the update U_{S2} has a greater numerical stability than U_{S1} if we do not include the quasi-Newton case. Another important practical result caused by performing the multiplier estimation by means of superlinear-order updates is the reduction of the difficulties associated with ill-conditioning, as they become unnecessary to increase the penalty parameter to the same extent as when the first-order update is used, and this makes the first more robust than the second.

REFERENCES

[1] Bertsekas, D. P. (1977). «Approximation procedures based on the method of multipliers». *J. Optim. Theory Appl.*, 23, 487-510.

[2] Bertsekas, D. P. (1982). *Constrained Optimization and Lagrange Multiplier Methods.* Academic Press, New York.

[3] Bertsekas, D. P. (1995). *Nonlinear Programming.* Athena Scientific, Belmont, Massachusetts.

[4] Bertsekas, D. P. (1998). *Network Optimization: Continuous and Discrete Models.* Athena Scientific, Belmont, Massachusetts.

[5] Brännlud, H., Sjelvgren, D. and Bubenko, J. A. (1988). «Short Term Generation Scheduling with Security Constraints». *IEEE Transactions on Power Systems*, 3, 1, 310-316.

[6] Conn, A. R., Gould, N. I. M. and Toint, Ph. L. (1991). «A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds». *SIAM J. Numer. Anal.*, 28(2), 545-572.

[7] Conn, A. R., Gould, N. I. M., Sartenaer, A. and Toint, Ph. L. (1996). «Convergence properties of an augmented Lagrangian algorithm for optimization with a combination of general equality and linear constraints». *SIAM J. Optim.*, 6(3), 674-703.

[8] Dembo, R. S., Eisenstat, S. C. and Steihaug, T. (1982). «Inexact Newton methods». *SIAM J. Numer. Anal.*, 19, 400-408.

[9] Dembo, R. S. (1987). «A primal truncated newton algorithm with application to large-scale nonlinear network optimization», *Math. Program. Stud.*, 31, 43-71.

[10] DIMACS (1991). *The first DIMACS international algorithm implementation challenge: The bench-mark experiments.* Technical Report, DIMACS, New Brunswick, NJ, USA.

[11] Gill, P. E. and Murray, W. (1979). «The computation of Lagrange-Multiplier estimates for constrained minimization». *Math. Program.*, 17, 32-60.

[12] Gill, P. E., Murray, W. and Wright, M. H. (1981). *Practical Optimization.* Academic Press, New York.

[13] Golub, G. H. and Van Loan, Ch. F. (1996). *Matrix computations.* (Third edition) Johns Hopkins, Baltimore and London.

[14] Griewank, A. and Toint, Ph. L. (1982). «Partitioned variable metric updates for large structured optimization problems». *Numer. Math.*, 39, 119-137.

[15] Heredia, F. J. and Nabona, N. (1991). *Large scale nonlinear network optimization with linear side constraints.* EURO XI, European Congress on Operational Research, Aachen, Germany, July.

[16] Heredia, F. J. and Nabona, N. (1995). «Optimum Short-Term Hydrothermal Scheduling with Spinning Reserve through Networks Flows». *IEEE Trans. on Power Systems*, 10(3), 1642-1651.

[17] Heredia, F. J. (1995). *Optimització de Fluxos No Lineals en Xarxes amb Constriccions a Banda. Aplicació a Models Acoblats de Coordinació Hidro-Tèrmica a Curt Termini.* Ph. D. Thesis, Statistics and Operations Research Dept., Universitat Politècnica de Catalunya, Barcelona.

[18] Hestenes, M. R. (1969). «Multiplier and gradient methods». *J. Optim. Theory Appl.*, 4, 303-320.

[19] Kennington, J. L. and Helgason, R. V. (1980). *Algorithms for network programming.* John Wiley and Sons, New York.

[20] Mijangos, E. (1997). *PFNRN03 user's guide.* Technical Report 97/05, Dept. of Statistics and Operations Research. Universitat Politècnica de Catalunya, Barcelona.

[21] Mijangos, E. and Nabona, N. (1996). *The application of the multipliers method in nonlinear network flows with side constraints.* Technical Report 96/10, Dept. of Statistics and Operations Research. Universitat Politècnica de Catalunya, Barcelona.

[22] Mijangos, E. and Nabona, N. (1998). «On the first-order estimation of multipliers from Kuhn-Tucker systems in multiplier methods using variable reduction». In: P. Kall and H.-J. Lüthi Editors, *Operations Research Proceedings 1998*, pp. 63-72. Springer-Verlag, Zürich.

[23] Mijangos, E. and Nabona, N. (1999). «On the compatibility of the classical first order multiplier estimates with the variable reduction techniques when there are nonlinear inequality constraints». *Qüestiio*, 23(1), 61-83.

[24] Mijangos, E. and Nabona, N. (1999). «On the first-order estimation of multipliers from Kuhn-Tucker systems». *Comput. Oper. Res.* To appear.

[25] Mijangos, E. (2000). *Superlinear-order multiplier estimation in nonlinear networks with nonlinear constraints.* EURO XVII, European Congress on Operational Research, Budapest, Hungary, July.

[26] Murtagh, B. A. and Saunders, M. A. (1978). «Large-scale linearly constrained optimization». *Math. Program.*, 14, 41-72.

[27] Murtagh, B. A. and Saunders, M. A. (1983-1998). *MINOS 5.5. User's guide.* Report SOL 83-20R, Department of Operations Research, Stanford University, Stanford, CA, USA, Revised July 1998.

[28] Nocedal, J. (1980). «Updating quasi-Newton matrices with limited storage». *Math. Comput.*, 35, 773-782.

[29] Powell, M. J. D. (1969). «A method for nonlinear constraints in minimization problems». In: *Optimization* (R. Fletcher, ed.), pp. 283-298. Academic Press, New York.

[30] Tapia, R. A. (1977). «Diagonalized multiplier methods and quasi-Newton methods for constrained optimization». *J. Optim. Theory. Appl.*, 22(2), 135-194.

[31] Toint, Ph. L. and Tuyttens, D. (1990). «On large scale nonlinear network optimization». *Math. Program.*, 48, 125-159.

[32] Wang, S. J., Shahidehpour, S. M., Kirschen, D. S., Mokhtari, S. and Irisarrri, G. D. (1994). *Short-Term Generation Scheduling with Transmission and Environement Constraints Using an Augmented Lagrangian Relaxation.* IEEE/PES Summer Meeting, San Francisco, CA, USA, 94 SM 572-8 PWRS.